CEWES MSRC/PET TR/99-07

Management Strategies for Scientific Data: Assessing Utility of HDF for CEWES MSRC Users

by

Dr. Alan M. Shih
Dr. M. Pauline Baker

04h00599

Work funded by the DoD High Performance Computing Modernization Program CEWES Major Shared Resource Center through

Programming Environment and Training (PET)

Supported by Contract Number: DAHC94-96-C0002
Nichols Research Corporation

Management Strategies for Scientific Data: Assessing Utility of HDF for CEWES MSRC Users

Dr. Alan M. Shih (ashih@ncsa.uiuc.edu)
Dr. M. Pauline Baker (baker@ncsa.uiuc.edu)
NCSA
University of Illinois at Urbana-Champaign

January 1999

Abstract

This paper reviews needs for data storage and management among high-performance computing users, particularly those at the CEWES MSRC. We also review the characteristics of the HDF file format package, with the intent of assessing its usefulness for MSRC users. Support packages for parallel I/O are also mentioned.

1 Introduction

High performance computing often generates very large data sets, measured in the gigabytes (GB, 10^9 bytes) or even terabytes (TB, 10^{12} bytes). Such large data sets pose significant problems for scientists in terms of storage, management, and extraction of meaningful information. Storage, management, and analysis practices must be improved to provide scalability. In this study, we review current practices for data management in the high-performance computing community. In particular, we look at data management needs and practices of users of the Department of Defense Major Shared Resource Center (MSRC) at the Army Corps of Engineers Waterways Experiment Station (CEWES) in Vicksburg, Mississippi. We also review a particular data format package, NCSA HDF (Hierarchical Data Format), and assess its utility for use at the MSRC. HDF is being used in NASA's Earth Observing System Data and Information System (EOSDIS) and in the DOE Accelerated Strategic Computing Initiative (ASCI) program, suggesting that it might be also appropriate for many users at the MSRC.

As a DoD MSRC, CEWES hosts many of the world's most advanced supercomputers and provides services to thousands of users nationwide. Many, if not most, of the MSRC users work on time-dependent scientific problems. For each time instance, up to several GBs of data are produced and saved. Tens, hundreds, or even thousands, of time steps are saved. For example, using CE-QUAL-IQM to study phenomena in the Chesapeake Bay generates more than 3 GB of data for a 365-day simulation. The research team routinely runs simulations for 10-year and 20-year runs. A blast-structure interaction study computed at the MSRC produced more than 1 TB of data. A store separation problem produces more than 25 GB of data with just 40 time steps. Similar data sets are produced by academic users. A computational fluid dynamics simulation on rotor-stator interaction conducted at the MSU/ERC has a potential to produce 50-100 GB of data for a complete simulation cycle. While by no means exhaustive, these examples provide a general picture of the profile of average users in today's high performance computing community.

2 Common Data Management Practices

2.1 Data Storage

Managing and accessing large data sets can become the most prominent problem for users of large data. Data storage and access techniques have not scaled with the increased performance of high-end computers, and many scientists still manage their data the same way they did years ago. This often involves ad hoc naming conventions, or somewhat more sophisticated schemes for cataloging data based on *metadata*, perhaps recording information about the data generation process or the content. Many of these systems are limited in their ability to grow and change with user access methods or sizes of databases.

2.2 Data Analysis

Post-processing simulation output requires retrieval from the data store. Many post-processing activities, such as visualization, are done on the user's local workstation. This usually requires:

1) Converting the data to be readable on the target machine
2) Transfering files between the mainframes and local clients via FTP
3) Extracting a subset from the data for analysis
4) Swapping various blocks of data on the local client as the entire data is too large for local system.

Thus, conventional approaches to post-processing of simulation output are limited by the local system capacity and network bandwidth. Obviously the user's ability to analyze the entire domain is compromised.

In the next section, we survey current packages and/or research efforts aimed at providing parts of the solution to management and analysis of scientific data. While projects such as those at the National Storage Laboratory at Lawrence Livermore National Laboratory promise to significantly increase the performance of traditional storage systems through hardware improvements, we focus here on software systems. We look both at packages for data storage formats and parallel I/O packages.

3 HDF

HDF is a multi-platform file format and collection of utilities for managing and manipulating scientific data. HDF supports file management in a heterogeneous collection of machines that might use varying word lengths and byte orders. HDF has been in use and development for over 10 years, and is widely used throughout the scientific and engineering research communities.

The current releases are HDF 4.1 and HDF 5.0. Version 5.0 represents a significant departure from the earlier model. Platforms supported include the IBM SP2, T3D and T3E, C90, SGI/CRAY Origin2000, SGI Power Challenge, SunOS, Sun Solaris, SGI IRIX, and Windows95/NT. The distribution consists of the HDF library, the HDF command line utilities, a test suite in source code, a Java Interface and the Java-based HDF Viewer (JHV).

HDF provides high-level interfaces for writing machine-independent data files. It provides APIs for JAVA, C, C++ and Fortran languages, and utilities for analyzing and converting HDF data files.

HDF currently supports several data structure types, including 8- and 24-bit raster images, color palettes, multi-dimensional arrays (Scientific Data Sets or SDS), and binary tables (Vdata). HDF provides means for storing, accessing, and manipulating relevant *metadata* along with the file. Rank, size, data generation details, etc. can all be stored as part of the HDF file and subsequently used to identify each data set and review characteristics.

HDF does not provide any 'standard' mechanism for storing unstructured data. However, many users have successfully built support for their unstructured data using the lower level HDF storage types.

As indicated below, there are several features of HDF that makes it attractive for data storage:

1. HDF is versatile. It supports several different data models. Each data model defines a specific aggregate data type and provides an API for reading, writing, and organizing data and metadata of the corresponding type.
2. HDF is self-describing. HDF allows an application to interpret the structure and contents of a file without any outside information.
3. HDF is flexible. It allows a user to mix and match related objects together in one file and then access them as a group or as individual objects by using "vgroups" feature in HDF. It is extensible, which means that it can easily accommodate new data models, regardless of whether they are added by the HDF development team or by HDF users. As noted above, this approach is taken by users with unstructured data to craft a data model suitable for their problem.
4. HDF is portable. An HDF file created on one computer can be read on a different system without modification. This is particularly important where users are generally working in a heterogeneous environment. The ability to share files across workstations and high-performance computers is crucial.

5. HDF is in wide use and HDF files can be read by many visualization packages, such as AVS and IBM Data Explorer.

HDF is developed at the National Center for Supercomputing Applications (NCSA). It is distributed in the public domain. Information about HDF, including a list of large user groups, is located at http://hdf.ncsa.uiuc.edu. HDF is being used in NASA's EOSDIS program (http://www.hq.nasa.gov/office/mtpe/ 97bi_rev/eosdis.htm) and in the DOE ASCI (http://www.llnl.gov/sccd/lc/asci) program. Current work on HDF includes efforts to support parallel I/O and modifications to the underlying data model to improve flexibility and internal consistency.

3.1 Extensions

Multi-file interface

The original model in HDF limited the user to working with one file at a time -- all read and write activity was to the currently open (unnamed) file. This extension allows a user to work with multiple files at once, referring to them by name. This gives programmers the same simple-to-use interface as HDF single-file interface for scientific data with the flexibility of multiple files. Through this extension, the user can open more file descriptors than system-defined limits allow.

Adaptive Mesh Refinement

Adaptive mesh refinement (AMR) is a technique for increasing the resolution of finite-difference codes by creating regions of refinement only around the areas that require it. This can dramatically reduce the amount of memory and CPU time required by a simulation compared to simply increasing the resolution of the problem as a whole. HDF can store AMR data. The AMR-HDF routines extend the SDS interface to store a grid hierarchy produced by AMR codes in the machine-independent HDF file format. The entire grid hierarchy is flattened so that it can be stored into a single file. Each grid in the hierarchy can be uniquely identified by its level, gridID, and time step so a user can also randomly access grids in this flattened file.

3.2 Related Data Formats

NetCDF

Developed at the Unidata Program Center in Boulder, Colorado, NetCDF (Network Common Data Format) is an interface for array-oriented data access and a library that provides an implementation of the interface. The NetCDF library also defines a machine-independent format for representing scientific data. It supports the creation, access, and sharing of scientific data. HDF supports the NetCDF calling interface. This means that applications that have been written to write and read NetCDF files can link to the HDF library and use HDF as the underlying storage mechanism without having to rewrite their code.

IEEEIO

IEEEIO is a compact library for storing multidimensional scientific data in a binary format. It is a machine-independent file format. IEEEIO depends on the IEEE 754 standard representations for floating point numbers to achieve its portability. It provides many of the capabilities of the HDF-SDS and NetCDF file formats however it is much smaller and simpler. In addition, it is reasonably fast when compared to raw F77 unformatted IO. The programming interface allows complete interoperability with the HDF and NetCDF formats if those libraries are linked. It is not intended to replace or compete with either HDF or NetCDF, but to provide an alternative when a simple approach is sufficient and when the extra functionality and size of these libraries is not needed or desired.

FlexIO

FlexIO is a compact API for storing multidimensional scientific data. It hides the differences between underlying file formats including HDF-SDS, IEEEIO, and network socket connections. It will also provide support for HDF5. It is designed to allow a user to use exactly the same subroutine calls to store scientific data regardless of the underlying file format. FlexIO includes C++, C, and Fortran77/F90 interfaces and it has been ported to Sun-Solaris, Digital Unix, Cray-Unicos, SGI-Irix (32 and 64 bit), Windows95 and NT.

FlexIO borrows its terminology and storage-style from HDF and NetCDF. The file stores a sequence of multi-dimensional arrays which are referred to generically as data sets. The dimensions and data-type are stored with each data set so that the data is completely self-describing. In addition, NetCDF-style named attributes can be stored with each data set to add information like coordinates, units, and other auxiliary information.

A set of higher level API's sit on top of FlexIO which permit simplified access to complex data structures like Finite-Element, Adaptive Mesh Refinement, and Unigrid data structures. In addition the MPIO interface provides access to parallel IO for MPI codes.

FlexIO supports IEEEIO binary file format, HDF 4.x SDS file format, and network socket remote file. It provides C, C++, and Fortran programming interfaces, as well as higher-level interfaces for complex data structures.

4 Parallel IO Support

Parallel computation has increased the computation power availability by utilizing multiple CPUs simultaneously and efficiently. This includes homogeneous and heterogeneous environment approaches. There are several research efforts addressing the parallel I/O issue that deserve attention.

Panda Project

Panda is an experimental scalable parallel I/O system being developed by the Center for Advanced Database Research at the University of Illinois. Panda uses the MPI message passing standard so it ports easily to different operating systems and can be used in a heterogeneous environment. The Panda project implements a very high level portable interface for writing scientific data sets on networks of workstations and parallel supercomputers. The Panda and HDF groups are working together under a grant funded by the Applied Information Systems Research Project to combine HDF machine independent data format with Panda's machine independent parallel IO for the NASA EOSDIS program. This will provide users with an easy-to-use cross platform way to integrate parallel I/O with their codes.

MPI-IO Project

MPI-IO is an extension to MPI for parallel I/O. It is being developed to fill the need for a portable parallel I/O interface for MPI programs. It has also been adopted by the Scalable I/O Initiative.

PPFS: Portable Parallel File System Project

PPFS is a system developed for experimenting with parallel file system designs. It enables the programmer to test different distributed caching techniques, data layouts, and pre-fetching strategies. There are three building blocks of the PPFS. They are:

Clients

A PPFS client is the integration of a user application node and the local caching, pre-fetching, and bookkeeping that enables an application to use the PPFS. Application-specific file system configuration and policies originate from the clients on behalf of the applications.

Servers

A PPFS server is an abstraction of an I/O device in the PPFS. It is built upon an underlying UNIX file system. These servers cache data, perform physical I/O, and pre-fetch data as specified by the clients.

Metadata

The PPFS metadata server is the persistent store of information that performs the bookkeeping duty and traces the whereabouts of data. It keeps track of which server contains which records of the file as well as bookkeeping information for different access modes which allows clients to share files that are kept in the metadata.

5 Implementation Issues

Adopting HDF, or any standard file format, as a supported file format for users at an MSRC site would provide a couple of obvious benefits. A base of expertise would quickly develop, which would help new members of the community adopt the standard. Data analysis tools, including visualization tools, could be provided that specifically addressed this format. As noted above, many off-the-shelf visualization tools, such as AVS and IBM DX, already read HDF. Further, data management schemes involving cataloging, search, and retrieval of HDF-stored files could be written and would apply to all HDF users at the MSRC.

In many cases, modifying codes to use HDF is not difficult. However, for commercial codes where souce is not provided, incorporating support for HDF would have to be done by the vendor. Users of such codes could still use HDF as their final storage format by using a translator to move from the vendor-supported format to HDF.

6 Summary

Data file sizes have grown dramatically during the last several years, partly thanks to the dramatic improvement of computer systems and storage capacity. Managing such enormously large data sets requires a new approach in data management. Otherwise, scientists simply can not extract the information contained in the data. Many scientists today are still managing their data the same way as they did years ago. A portable and efficient data format is needed to allow scientists to store their data and metadata in such a way that they can access the data easily and efficiently.

7 References

1. Sumpter, R. M., "Whitepaper on Data Management," Lawrence Livermore National Laboratory, February 10, 1994, Version 1.0.
2. Grossman, R. L., Hanley, D., and Bailey S., "A Tutorial on High Performance Data Management Using PTool ," Laboratory for Advanced Computing, University of Illinois at Chicago, March, 1994.
3. Bordawekar, R. et. al., "A Model and Compilation Strategy for Out-of-Core Data Parallel Programs," NPAC, Syracuse University, June 1996.

8 World Wide Web References

HDF Home Page:	http://hdf.ncsa.uiuc.edu
HDF Utilities:	http://hdf.ncsa.uiuc.edu/tools.html
HDF Info Page:	http://www.hdfinfo.com/
HDF-SDS:	
HDF-SDS Home Page:	http://bach/IO/SDSlibrary.html
IEEEIO Home Page:	http://bach/IEEEIO/IEEEIO.html
IEEEIO Utilities:	http://bach/IEEEIO/Utilities.html
FlexIO Home Page:	http://bach/IEEEIO/
NetCDF Home Page:	http://www.unidata.ucar.edu/packages/netcdf/
Parallel IO Home Page:	http://www.llnl.gov/sccd/lc/piop/
MPI IO Home Page:	http://parallel.nas.nasa.gov/MPI-IO/